QUICR-learning for Multi-Agent Coordination

Adrian K. Agogino
UCSC, NASA Ames Research Center
Mailstop 269-3
Moffett Field, CA 94035
adrian@email.arc.nasa.gov

Kagan Tumer
NASA Ames Research Center
Mailstop 269-3
Moffett Field, CA 94035
kagan@email.arc.nasa.gov

Abstract

Coordinating multiple agents that need to perform a sequence of actions to maximize a system level reward requires solving two distinct credit assignment problems. First, credit must be assigned for an action taken at time step t that results in a reward at time step $t' > t$. Second, credit must be assigned for the contribution of agent i to the overall system performance. The first credit assignment problem is typically addressed with temporal difference methods such as Q-learning. The second credit assignment problem is typically addressed by creating custom reward functions. To address both credit assignment problems simultaneously, we propose the "Q Updates with Immediate Counterfactual Rewards-learning" (QUICR-learning) designed to improve both the convergence properties and performance of Q-learning in large multi-agent problems. QUICR-learning is based on previous work on single-time-step counterfactual rewards described by the collectives framework. Results on a traffic congestion problem shows that QUICR-learning is significantly better than a Q-learner using collectives-based (single-time-step counterfactual) rewards. In addition QUICR-learning provides significant gains over conventional and local Q-learning. Additional results on a multi-agent grid-world problem show that the improvements due to QUICR-learning are not domain specific and can provide up to a ten fold increase in performance over existing methods.

Introduction

Coordinating a set of interacting agents that take sequences of actions to maximize a system level performance criteria is a difficult problem. Addressing this problem with a large single agent reinforcement learning algorithm is ineffective in general because the state-space becomes prohibitively large. A more promising approach is to give each agent in the multi-agent system its own reinforcement learner. This approach, however, introduces a new problem: how to assign credit for the contribution of an agent to the system performance, which in general is a function of all agents. Allowing each agent to try to maximize the system level global reward is problematic in all but the smallest problems as an agent's reward is masked by the actions of all the other agents in the system. In Markov Decisions Problems (MDPs) presented in this paper, the global reward may be influenced by as many as 800 actions (actions of 40 agents over 20 time steps). Purely local rewards allow us to overcome this "signal-to-noise" problem. On the other hand, local rewards are problematic, since there are no guarantees that policies formed by agents that maximize their local reward will also maximize the global reward.

In this paper, we present "Q Updates with Immediate Counterfactual Rewards learning" (QUICR-learning) which uses agent-specific rewards that ensure fast convergence in multi-agent coordination domains. Rewards in QUICR-learning are both heavily *agent-sensitive*, making the learning task easier, and *aligned* with the system level goal, ensuring that agents receiving high rewards are helping the system as a whole. QUICR-learning uses standard temporal difference methods but because of its unique reward structure, provides significantly faster convergence than standard Q-learning in large multi-agent systems. In the next Section, we present a brief summary of the related research. We then discuss the temporal and structural credit assignment problems in multi-agent systems, and describe the QUICR-learning algorithm. The following two sections present results on two different problems that require coordination, showing that QUICR-learning performs up to ten times better than standard Q-learning in multi-agent coordination problems. Finally we discuss the implications and limitations of QUICR-learning and highlight future research directions.

Previous Work

Currently the best multi-agent learning algorithms used in coordinating agents address the structural credit assignment problem by leveraging domain knowledge. In robotic soccer for example, player specific subtasks are used, followed by tiling to provide good convergence properties (Stone, Sutton, & Kuhlmann 2005). In a robot coordination problem for the foraging domain, specific rules induce good division of labor (Jones & Mataric 2003). In domains where groups of agents can be assumed to be independent, the task can be decomposed by learning a set basis functions used to represent the value function, where each basis only processes a small number of the state variables (Guestrin, Lagoudakis, & Parr 2002). Also multi-agent Partially Observable Markov Decision Precesses (POMDPs) can be simplified through piecewise linear rewards (Nair *et al.* 2003). There have also

been several approaches to optimizing Q-learning in multi-agent systems that do not use independence assumptions. For a small number of agents, game theoretic techniques were shown to lead to a multi-agent Q-learning algorithm proven to converge (Hu & Wellman 1998). In addition, the equivalence between structural and temporal credit assignment was shown in (Agogino & Tumer 2004), and methods based on Bayesian methods were shown to improve multi-agent learning by providing better exploration capabilities (Chalkiadakis & Boutilier 2003). Finally, task decomposition in single agent RL can be achieved using hierarchical reinforcement learning methods such as MAXQ value function decomposition (Dietterich 2000).

Credit Assignment Problem

The multi-agent temporal credit assignment problem consists of determining how to assign rewards (e.g., credit) for a sequence of actions. Starting from the current time step t, the undiscounted sum of rewards till a final time step T can be represented by:

$$R_t(s_t(a)) = \sum_{k=0}^{T-t} r_{t+k}(s_{t+k}(a)) \; . \qquad (1)$$

where a is a vector containing the actions of all agents at all time steps, $s_t(a)$ is the state function returning the state of *all* agents for a single time step, and $r_t(s)$ is the single-time-step reward function, which is a function of the states of *all* of the agents.

This reward is a function of all of the previous actions of all of the agents. Every reward is a function of the states of all the agents, and every state is a function of all the actions that preceded it (even though it is Markovian, the previous states ultimately depend on previous actions). In a system with n agents, a reward received on the last time step can be affected by up to $n * T$ actions. Looking at rewards received at different time steps, on average $\frac{1}{2}n * T$ actions may affect a reward in tightly coupled systems. Agents need to use this reward to evaluate their single action; in the domains presented in the results sections with forty agents and twenty time steps there are up to 800 actions affecting the reward!

Standard Q-Learning

Reinforcement learners such as Q-learning address (though imperfectly) how to assign credit of future rewards to an agent's current action. The goal of Q-learning is to create a policy that maximizes the sum of future rewards, $R_t(s_t(a))$, from the current state (Kaelbling, Littman, & Moore 1996; Sutton & Barto 1998; Watkins & Dayan 1992). It does this by maintaining tables of Q-values, which estimate the expected sum of future rewards for a particular action in a particular state. In the TD(0) version of Q-learning, a Q-value, $Q(s_t, a_t)$, is updated with the following Q-learning rule [1]:

$$Q(s_t, a_t) = r_t + max_a Q(s_{t+1}, a) \; . \qquad (2)$$

[1]To simplify notation, this paper uses Q-learning update notation for deterministic (where learning rate $\alpha = 1$ converges), undiscounted Q-learning. The extensions to non-deterministic and discounted cases through the addition of learning rate and discounting parameters are straight-forward.

This update assumes that the action a_t is most responsible for the immediate reward r_t, and is less responsible for the sum of future rewards, $\sum_{k=1}^{T-t} r_{t+k}(s_{t+k}(a))$. This assumption is reasonable since rewards in the future are affected by uncertain future actions and noise in state transitions. Instead of using the sum of future rewards directly to update its table, Q-learning uses a Q-value from the next state entered as an estimate for those future rewards. Under certain assumptions, Q-values are shown to converge to the actual value of the future rewards (Watkins & Dayan 1992).

Even though Q-learning addresses the temporal credit assignment problem (i.e., tries to apportion the effects of all actions taken at other time steps to the current reward), it does not address the structural credit assignment problem (i.e., how to apportion credit to the individual agents in the system). As a result when many agents need to coordinate their actions, standard Q-learning is generally slow since it needs all agents to tease out their time dependent contribution to the global system performance based on the global reward they receive. An additional issue with standard Q-learning is that in general an agent needs to fully observe the actions of other agents in order to compute its reward. This requirement is reduced in the "Local Q-Learning" and "QUICR-Learning" algorithms presented in this paper.

Local Q-Learning

One way to address the structural credit assignment problem and allow for fast learning is to assume that agents' actions are independent. Without this assumption, the immediate reward function for a multi-agent reward system may be a function of all the states:

$$r_t(s_{t,1}(a_1), s_{t,2}(a_1), \ldots, s_{t,n}(a_n)) \; ,$$

where $s_{t,i}(a_i)$ is the state for agent i and is a function of only agent i's previous actions. The number of states determining the reward grows linearly with the number of agents, while the number of actions that determine each state grows linearly with the number of time steps. To reduce the huge number of actions that affect this reward, often the reward is assumed to be linearly separable:

$$r_t(s_t) = \sum_i w_i r_{t,i}(s_{t,i}(a_i)) \; .$$

Then each agent receives a reward $r_{t,i}$ which is only a function of its action. Q-learning is then used to resolve the remaining temporal credit assignment problem. If the agents are indeed independent and their pursuing their local objectives has no deleterious side effects on each other, this method leads to a significant speedup in learning rates as an agent receives direct credit for its actions. However, if the agents are coupled, then though local Q-learning will allow fast convergence, the agents will tend to converge to the wrong policies (i.e., policies that are not *globally* desirable). In the worst case of strong agent coupling, this can lead to worse than random performance (Wolpert & Tumer 2001)).

QUICR-Learning

In this section we present QUICR-learning, a learning algorithm for multi-agent systems that does not assume that

the system reward function is linearly separable. Instead it uses a mechanism for creating rewards that are a function of all of the agents, but still provide many of the benefits of hand-crafted rewards. In particular, QUICR-learning rewards have:

1. high "alignment" with the overall learning task.
2. high "sensitivity" to the actions of the agent.

The first property of alignment means that when an agent maximizes its own reward it tends to maximize the overall system reward. Without this property, a large multi-agent system can lead to agents performing useless work, or worse, working at cross-purposes. Having aligned rewards is critical to multi-agent coordination. Reward sensitivity means that an agent's reward is more sensitive to its own actions than to other agents' actions. This property is important for agents to learn quickly. Note that assigning the full system reward to all the agents (e.g., standard Q-learning) has low agent-sensitivity, since each agent's reward depends on the actions of all the other agents.

QUICR-learning is based on providing agents with rewards that are both aligned with the system goals and sensitive to the agent's states. It aims to provide the benefits of customizing rewards without requiring detailed domain knowledge. In a task where the reward can be expressed as in Equation 1, let us introduce the difference reward (adapted from (Wolpert & Tumer 2001)) given by:

$$D_t^i(s_t(a)) \quad = \quad R_t(s_t(a)) - R_t(s_t(a - a_{t,i})) \; ,$$

where $a - a_{t,i}$ denotes a counterfactual state where agent i has not taken the action it took in time step t (e.g., the action of agent i has been removed from the vector containing the actions of all the agents before the system state has been computed). Decomposing further, we obtain:

$$\begin{aligned} D_t^i(s_t(a)) &= \sum_{k=0}^{T-t} r_{t+k}(s_{t+k}(a)) - r_{t+k}(s_{t+k}(a - a_{t,i})) \\ &= \sum_{k=0}^{T-t} d_{t+k}(s_{t+k}(a), s_{t+k}(a - a_{t,i})) \; . \end{aligned} \quad (3)$$

where $d_t(s_1, s_2) = r_t(s_1) - r_t(s_2)$. (We introduce the single time step "difference" reward d_t to keep the parallel between Equations 1 and 3). This reward is more sensitive to an agent's action than r_t since much of the effects of the other agents are subtracted out with the counterfactual (Wolpert & Tumer 2001). In addition often an agent does not need to fully observe the actions of other agents to compute the difference reward, since in many domains the subtraction cancels out many of the variables. Unfortunately in general $d_t(s_1, s_2)$ is non-Markovian since the second parameter may depend on previous states, making its use troublesome in a learning task involving both a temporal and structural credit assignment. (This difficulty is examined further below.)

In order to overcome this shortcoming of Equation 3, let us make the following assumption:

1. The counterfactual action $a - a_{t,i}$ moves agent i to an absorbing state, s_b.
2. s_b is independent of the agent's current (or previous) state(s).

These assumptions are necessary to have the Q-table backups approximate the full time-extended difference reward given in equation 3. Forcing the counterfactual action $a - a_{t,i}$ to move the agent into an absorbing state is necessary to enable the computation of equation 3. Without this assumption the ramifications of $a - a_{t,i}$ would have to be forward propagated through time to compute D_t^i. The second assumption is necessary to satisfy the Markov property of the system in a subtle way. While the next state s_{t+1} caused by action a_t is generally a function of the current state, the absorbing state s_b caused by the action $a - a_{t,i}$ should be independent of the current state in order for the Q-table updates to propagate correctly. The problem here is that Q-table backups are based on the next state entered, not the counterfactual state entered. The experimental results sections later in this paper show examples of the problems caused when this assumption is broken. In addition to these assumptions, the state of an agent should not be a direct function of the actions of other agents, otherwise we would have to compute the effects of counterfactual action $a - a_{t,i}$ on all the the agents. However, this does not mean that the agents in the system are independent. They still strongly influence each other through the system reward, which in general is nonlinear.

Given these conditions, the counterfactual state for time $t + k$ is computed from the actual state at time $t + k$, by replacing the state of agent i at time t with s_b. Now the difference reward can be made into a Markovian function:

$$d_t^i(s_t) = r_t(s_t) - r_t(s_t - s_{t,i} + s_b) \; , \quad (4)$$

where the expression $s_t - s_{t,i} + s_b$ denotes replacing agent i's state with state s_b.

Now the Q-learning rule can be applied to the difference reward, resulting in the QUICR-learning rule:

$$\begin{aligned} Q_{UICR}(s_t, a_t) &= r_t(s_t) - r_t(s_t - s_{t,i} + s_b) \\ &\quad + max_a Q(s_{t+1}, a) \\ &= d_t^i(s_t) + max_a Q(s_{t+1}, a) \; . \end{aligned} \quad (5)$$

Note that since this learning rule *is* Q-learning, albeit applied to a different reward structure, it shares all the convergence properties of Q-learning. In order to show that Equation 5 leads to good system level behavior, we need to show that agent i maximizing $d_t^i(s_t)$ (e.g., following Equation 5) will maximize the system reward r_t. Note that by definition $(s_t - s_{t,i} + s_b)$ is independent of the actions of agent i, since it is formed by moving agent i to the absorbing state s_b from which it cannot emerge. This effectively means the partial differential of $d_t^i(s_t)$ with respect to agent i is[2]:

$$\frac{\partial}{\partial_{s_i}} d_t^i(s_t) \quad = \quad \frac{\partial}{\partial_{s_i}} (r_t(s_t) - r_t(s_t - s_{t,i} + s_b))$$

[2]Though in this work we show this result for differentiable states, the principle applies to more general states, including discrete states.

$$= \frac{\partial}{\partial_{s_i}} r_t(s_t) - \frac{\partial}{\partial_{s_i}} r_t(s_t - s_{t,i} + s_b)$$
$$= \frac{\partial}{\partial_{s_i}} r_t(s_t) - 0$$
$$= \frac{\partial}{\partial_{s_i}} r_t(s_t). \quad (6)$$

Therefore any agent i using a learning algorithm to optimize $d_t^i(s_t)$ will tend to optimize $r_t(s_t)$.

QUICR-Learning and WLU

The difference reward in QUICR-Learning, $d_t^i(s_t) = r_t(s_t) - r_t(s_t - s_{t,i} + s_b)$, is closely related to the "Wonderful Life Utility" used in multiple non-time-extended problems (Wolpert & Tumer 2001):

$$WLU^i(s) = r(s) - r(s - s_i + c)\ , \quad (7)$$

where c is independent of state s_i. The strait-forward conversion of this into single-time-step rewards is:

$$WLU_t^i(s_t) = r(s_t) - r(s_t - s_{t,i} + c_t)\ , \quad (8)$$

where c_t is independent of state s_t. The reward $d_t^i(s_t)$ is a form of $WLU_t^i(s_t)$ that places greater restriction on c_t: it must be independent of *all* previous states and should be an absorbing state. Without these restrictions WLU_t^i creates problems with reward alignment and sensitivity. Without the restrictions, WLU_t^i is aligned with the system reward for single-time-step problems since c_t is independent of the agent's current state. The subtle difficulty is that values of WLU_t^i get propagated back to previous states through Q-learning. If c_t is not independent of all previous states, values that are not aligned with the system reward may be propagated back to previous states. While these differences sometimes do not matter, experimental results presented later in this paper show that they are often important. Having c_t be an absorbing state (as done in QUICR-learning) is needed to keep the learners' Q values approximate the time extended difference reward $D_t^i(s_t(a))$.

Traffic Congestion Experiment

To evaluate the performance of QUICR-learning, we perform experiments that test the ability of agents to maximize a reward based on an abstract traffic simulation. In this experiment n drivers can take a combination of m roads to make it to their destination. Each road j has an ideal capacity c_j representing the size of the road. In addition each road has a weighting value w_j representing a driver's benefit from taking the road. This weighting value can be used to represent such properties such as a road's difficulty to drive on and convenience to destination. In this experiment a driver starts on a road chosen randomly. At every time step, the driver can choose to stay on the same road or to transfer to one of two adjacent roads. In order to test the ability of learners to perform long term planning, the global reward is zero for all time steps, except for the last time step when it is computed as follows:

$$r_t = \sum_j k_{j,t} e^{(-\frac{k_{j,t}}{c_j})}\ , \quad (9)$$

where $k_{j,t}$ is the number of drivers on road j at time t.

Learning Algorithms

In our experiments for both this traffic congestion problem and the grid world problem (presented in the next section) we tested the multi-agent system using variants of the temporal difference method with $\lambda = 0$ (TD(0)). The actions of the agents were chosen using an epsilon-greedy exploration scheme and tables were initially set to zero with ties broken randomly (in the traffic congestion experiment ϵ was set to 0.05 and in the multi-agent grid world experiment ϵ was set to 0.15). In this case, there were 60 agents taking actions for six consecutive time steps. The learning rate was set to 0.5 (however to simplify notation we do not show the learning rates in the update equations). The four algorithms are as follows:

- Standard Q-learning is based on the full reward r_t:
$$Q(s_t, a_t) = r_t(s_t) + max_a Q(s_{t+1}, a)\ . \quad (10)$$
- Local Q-learning is only a function of the specific driver's own road, j:
$$Q_{loc}(s_t, a_t) = k_{j,t} e^{(-\frac{k_{j,t}}{c_j})} + max_a Q_{loc}(s_{t+1}, a)\ .$$
- QUICR-learning instead updates with a reward that is a function of all of the states, but uses counterfactuals to suppress the effect of other driver's actions:
$$Q_{UICR}(s_t, a_t) = r_t(s_t) - r_t(s_t - s_{t,i} + s_b) + max_a Q_{UICR}(s_{t+1}, a)\ ,$$
where $s_t - s_{t,i} + s_b$ is the state resulting from removing agent $i's$ state and replacing it with the absorbing state s_b.
- WLU_t Q-learning is similar to QUICR-learning, but uses a simpler form of counterfactual state. Instead of replacing the state $s_{t,i}$ by the absorbing state s_b, it is replaced by the state that the driver would have been in if he had taken action 0, which in this case is the same road he was on the previous time step : $s_{t-1,i}$. The resulting update equation is:
$$Q_{WLU}(s_t, a_t) = r_t(s_t) - r_t(s_t - s_{t,i} + s_{t-1,i}) + max_a Q_{WLU}(s_{t+1}, a)\ .$$

Results

Experimental results on the traffic congestion problem show that QUICR-learning learns more quickly and achieves a higher level of performance than the other learning methods (Figure 1). While standard Q-learning is able to improve performance with time, it learns very slowly. This slow learning speed is caused by Q-learning's use of the full reward $r_t(s_t)$, which is a function of the actions of all the other drivers. When a driver takes an action that is beneficial, the driver may still receive a poor reward if some of the fifty nine other drivers took poor actions at the same time. In contrast, local Q-learning learns quickly, but since it uses a reward that is not aligned with the system reward, drivers using local Q-learning eventually learn to take bad actions. Early in learning drivers using local Q-learning perform well as they learn to use the roads with high capacity and higher

weighting. However, as learning progresses the drivers start overusing the roads with high weighting, since their reward does not take into account that using other roads would benefit the system as a whole. This system creates a classic Tragedy of the Commons scenario. By over utilizing the "beneficial" roads, the drivers end up being worse off than if they had acted in a cooperative manner. Drivers using WLU_t Q-learning have similar problems because although they are aligned at each time step, they are not aligned across time steps.

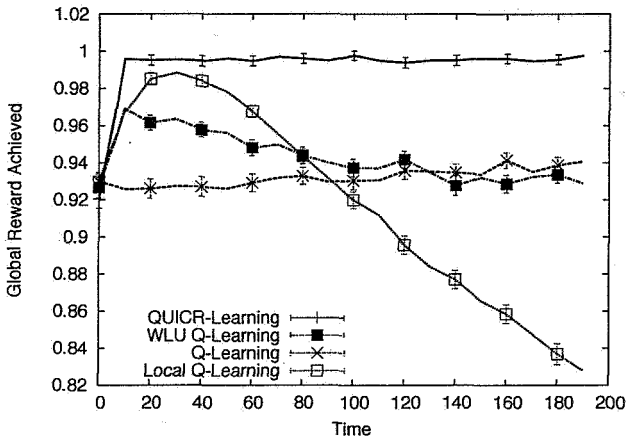

Figure 1: Traffic Congestion Problem (60 Agents).

Multi-agent Grid World Experiment

The second set of experiments we conducted involved a standard grid world problem (Sutton & Barto 1998). In this problem, at each time step, the agent can move up, down, right or left one grid square, and receives a reward (possibly zero) after each move. The observable state space for the agent is its grid coordinate and the reward it receives depends on the grid square to which it moves. In the episodic version, which is the focus of this paper, the agent moves for a fixed number of time steps, and then is returned to its starting location.

In the multi-agent version of the problem there are multiple agents navigating the grid simultaneously influencing each others' rewards. In this problem agents are rewarded for observing tokens located in the grid. Each token has a value between zero and one, and each grid square can have at most one token. When an agent moves into a grid square, it observes a token and receives a reward for the value of the token. Rewards are only received on the first observation of the token. Future observations from the agent or other agents do not receive rewards in the same episode. More precisely, r_t is computed by:

$$r_t(s_t) = \sum_i \sum_j V_j I^t_{s_{t,i}=L_j} , \qquad (11)$$

where I^t is the indicator function which returns one when an agent in state $s_{t,i}$ is in the location of an unobserved token L_j. The global objective of the multi-agent grid world problem is to observe the highest aggregated value of tokens in a fixed number of time steps T.

Learning Algorithms

As in the traffic congestion problem, we test the performance of the following four learning methods:

- Standard Q-learning is based on the full reward r_t:

$$Q(s_t, a_t) = r_t(s_t) + max_a Q(s_{t+1}, a) . \qquad (12)$$

- Local Q-learning is only a function of the specific agent's own state:

$$Q_{loc}(s_t, a_t) = sum_j V_j I^{t,i}_{s_{t,i}=L_j} + max_a Q_{loc}(s_{t+1}, a).$$

- QUICR-learning instead updates with a reward that is a function of all of the states, but uses counterfactuals to suppress the effect of other agents' actions:

$$Q_{UICR}(s_t, a_t) = r_t(s_t) - r_t(s_t - s_{t,i} + s_b) + max_a Q_{UICR}(s_{t+1}, a) ,$$

where $s_t - s_{t,i} + s_b$ is the state resulting from removing agent $i's$ state and replacing it with the absorbing state s_b.

- WLU Q-learning is similar to QUICR-learning, but uses a different counterfactual state. Instead of replacing the state $s_{t,i}$ by the absorbing state s_b, it is replaced by the state that the agent would have been in if he had taken action 0, which causes the agent to move to the right:

$$Q_{WLU}(s_t, a_t) = r_t(s_t) - r_t(s_t - s_{t,i} + s^{right}_{t-1,i}) + max_a Q_{WLU}(s_{t+1}, a) ,$$

where $s^{right}_{t-1,i}$ is the state to the right of $s_{t-1,i}$.

Results

In this experiment we use a token distribution where the "highly valued" tokens are concentrated in one corner, with a second concentration near the center where the rovers are initially located. This experiment is described in more detail in (Tumer, Agogino, & Wolpert 2002).

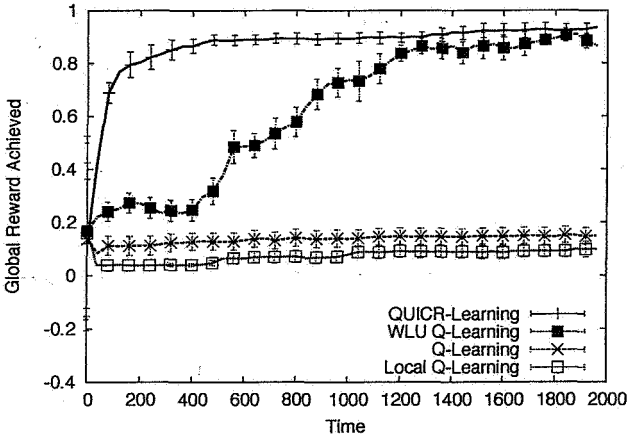

Figure 2: Multi-Agent Grid World Problem (40 Agents).

Figure 2 shows the performance for 40 agents on a 400 unit-square world for episodes of 20 time steps (error bars of $\pm$ one σ are included). The performance measure in these figures is the sum of full rewards ($r_t(s_t)$) received in an episode, normalized so that the maximum reward achievable is 1.0. Note all learning methods are evaluated on the same reward function, independent of the reward function that they are internally using to assign credit to the agents.

The results show that local Q-learning generally produces poor results. This problem is caused by all agents aiming to acquire the most valuable tokens, and congregating towards the corner of the world where such tokens are located. In essence, in this case agents using local Q-learning compete, rather than cooperate. The agents using standard Q-learning do not fare better, as the agents are plagued by the credit assignment problem associated with each agent receiving the full world reward for each individual action they take. Agents using QUICR-learning on the other hand learn rapidly, outperforming both local and standard Q-learning by a factor of six (over random rovers). Agents using WLU_t Q-learning eventually achieve high performance, but learn three times more slowly than agents using QUICR-learning.

Discussion

Using Q-learning to learn a control policy for a single agent in a coordination problem with many agents is difficult, because an agent will often have little influence over the reward it is trying to maximize. In our examples, an agent's reward received after an action could be influenced by as many as 800 other actions from other time-steps and other agents. Even temporal difference methods that perform well in single agent systems will be overwhelmed by the number of actions influencing a reward in the multi-agent setting. To address this problem, this paper introduces QUICR-learning, which aims at reducing the impact of other agent's actions without assuming linearly separable reward functions. Within the Q-learning framework, QUICR-learning uses the difference reward computed with immediate counterfactuals. While eliminating much of the influence of other agents, this reward is shown mathematically to be aligned with the global reward: agents maximizing the difference reward will also be maximizing the global reward. Experimental results in a traffic congestion problem and a grid world problem confirm the analysis, showing that QUICR-learning learns in less time than standard Q-learning, and achieves better results than Q-learning variants that use local rewards and assume linear separability. While this method was used with TD(0) Q-learning updates, it also extends to TD(λ), Sarsa-learning and Monte Carlo estimation.

In our experiments an agent's state is never directly influenced by the actions of other agents. Despite this, the agents are still tightly coupled by virtue of their reward. Agents can, and did affect each other's ability to achieve high rewards, adding complexity that does not exist in systems where agents are independent. In addition even though agents do not directly influence each other's states, they indirectly affect each other through learning: an agent's actions can impact another agent's reward, and the agents select actions based on previous rewards received. Hence an agent's action at time step t does affect other agents at $t' > t$ through their learning algorithms. The mathematics in this paper does not address these indirect influences and is a subject of further research. However, experimental evidence shows that agents can still cooperate despite these indirect effects. In fact even when agents directly influence each other's states, in practice they may still cooperate effectively as long as they use agent-sensitive rewards that are aligned with the system reward as has been shown in experiments presented in (Agogino & Tumer 2004) .

References

Agogino, A., and Tumer, K. 2004. Unifying temporal and structural credit assignment problems. In *Proc. of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems*.

Chalkiadakis, G., and Boutilier, C. 2003. Coordination in multiagent reinforcement learning: A bayesian approach. In *Proc. of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*.

Dietterich, T. G. 2000. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence* 13:227–303.

Guestrin, C.; Lagoudakis, M.; and Parr, R. 2002. Coordinated reinforcement learning. In *Proceedings of the 19th International Conference on Machine Learning*.

Hu, J., and Wellman, M. P. 1998. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, 242–250.

Jones, C., and Mataric, M. J. 2003. Adaptive division of labor in large-scale multi-robot systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-03)*, 1969–1974.

Kaelbling, L. P.; Littman, M. L.; and Moore, A. W. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 4:237–285.

Nair, R.; Tambe, M.; Yokoo, M.; Pynadath, D.; and Marsella, S. 2003. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*.

Stone, P.; Sutton, R. S.; and Kuhlmann, G. 2005. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior*.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Tumer, K.; Agogino, A.; and Wolpert, D. 2002. Learning sequences of actions in collectives of autonomous agents. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 378–385.

Watkins, C., and Dayan, P. 1992. Q-learning. *Machine Learning* 8(3/4):279–292.

Wolpert, D. H., and Tumer, K. 2001. Optimal payoff functions for members of collectives. *Advances in Complex Systems* 4(2/3):265–279.